Increased Resilience in Threshold Cryptography: Sharing a Secret with Devices That Cannot Store Shares*

Koen Simoens, Roel Peeters, and Bart Preneel

Department of Electrical Engineering – COSIC
Katholieke Universiteit Leuven and IBBT, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. Threshold cryptography increases security and resilience by sharing a private cryptographic key over different devices. Many personal devices, however, are not suited for threshold schemes, because they do not offer secure storage, which is needed to store shares of the private key. We present a solution that allows to include devices without them having to store their share. Shares are stored in protected form, possibly externally, which makes our solution suitable for low-cost devices with a factory-embedded key, e.g., car keys and access cards. By using pairings we achieve public verifiability in a wide range of protocols, which removes the need for private channels. We demonstrate how to modify existing discrete-log based threshold schemes to work in this setting. Our core result is a new publicly verifiable distributed key generation protocol that is provably secure against static adversaries and does not require all devices to be present.

1 Introduction

The increased capabilities of mobile devices and connectivity with the rest of the world have made the use of these devices exceed their original purpose. Mobile phones are being used to read e-mail, authorise bank transactions or access social network sites. As a consequence, personal devices are used more and more for security-sensitive tasks. Moreover, personal data are copied to these devices and need to be protected. In both cases, by using cryptography, security reduces to the management of cryptographic keys. Although mobility is considered as a major benefit, it is a weakness in terms of security and reliability. Mobile devices are susceptible to theft, can easily be forgotten or lost, or run out of battery power. These issues can be mitigated by introducing threshold cryptography.

The aim of threshold cryptography is to protect a key by sharing it amongst a number of entities in such a way that only a subset of minimal size, namely the threshold $t + 1$, can use the key. No information about the key can be learnt from t or less shares. The setup of a threshold scheme typically involves a Distributed Key Generation (DKG) protocol. In a DKG protocol a group of

* The original publication is available at http://www.springerlink.com

entities cooperate to jointly generate a key pair and obtain shares of the private key. These shares can then be used to sign or decrypt on behalf of the group.

The benefits of a threshold scheme are increased security, because an adversary can compromise up to t devices, and resilience, since any subset of $t + 1$ devices is sufficient. To increase resilience we want to maximise the number of devices included in the threshold scheme. However, the number of personal devices suitable for threshold schemes is limited because many of these do not incorporate secure storage, which is needed to store shares of the private key. We enlarge the group of high-end devices by also considering small devices with public-key functionality, e.g., car keys or access cards. Typically, these small devices have a factory-embedded private key, which cannot be updated and is the only object that resides in tamper-proof secure storage.

Our proposed solution allows to store shares in protected form,[1] possibly externally. These protected shares are generated through a run of our new DKG protocol. By using pairings we achieve publicly verifiability, which implies that the correctness of any device's contribution can be verified by any entity observing the DKG protocol thus eliminating private channels. As such, not every device needs to be present during the DKG protocol. We apply our setting to existing threshold schemes and we show how shares can be used implicitly without being needed in unprotected form. Furthermore, some devices can be completely ignorant of the underlying schemes and only serve as partial decryption oracles.

Organisation. Related work is surveyed in Sect. 2. In Sect. 3 we introduce some basic concepts. We give an overview of typical routines in a threshold setting and we describe our communication and adversarial model. Security definitions are given along with an overview of relevant number-theoretic assumptions and notation on bilinear pairings. In Sect. 4 we present how to protect shares and our main result, which is a new publicly verifiable DKG protocol that does not require every device to be present. In Sect. 5 we demonstrate how protected shares can easily be used in discrete-log based cryptosystems and signature schemes. More specifically, we demonstrate this for the ElGamal [14] and the Cramer-Shoup [7] cryptosystems, and the Schnorr signature scheme [25].

2 Related Work

Shamir's early idea [27] of distributing shares of a secret as evaluations of a polynomial has become a standard building block in threshold cryptography. Feldman [8] introduced verifiable secret sharing (VSS) by publishing the coefficients of this polynomial hidden in the exponent of the generator of a group in which the discrete-log assumption holds. Pedersen [22] then used this idea to construct the first distributed key generation (DKG) protocol, sometimes referred to as Joint Feldman, by having each player in a group run an instance of Feldman's

[1] An obvious answer would be to encrypt shares under the devices' public keys. This is undesired because at some point shares will be in the clear in unprotected memory.

protocol in parallel. Soon thereafter, Pedersen [23] produced another remarkable result. He made Feldman's VSS scheme information-theoretically secure by choosing two polynomials and broadcasting the corresponding coefficients as paired commitments, which are known as Pedersen commitments. Gennaro et al. [15] pointed out that the uniformity of the key produced by Pedersen's DKG protocol cannot be guaranteed in the context of a rushing adversary. They constructed a new DKG protocol [15] by first running Pedersen's VSS in parallel (Joint Pedersen). Since Pedersen VSS does not produce a public key, an extra round of communication, basically an instance of Joint Feldman on the first polynomial, has to be added to compute the public key. They proved their protocol secure against a static adversary by means of a simulation argument. Interestingly, Gennaro et al. showed later [16] that, despite the biased distribution of the key, certain discrete-log schemes that use Pedersen DKG can still be proved secure at the cost of an increased security parameter. Canetti et al. [6] used interactive knowledge proofs and erasures, i.e., players erase private data before commitments or public values are broadcast, in the key construction phase of the DKG of [15] to make the protocol secure against adaptive adversaries. Comparable adaptively secure threshold schemes were presented by Frankel et al. [10].

In the protocols discussed so far, it is assumed that there are private channels between each pair of players. Both [6] and [10] suggest that these channels can still be established even with an adaptive adversary using the non-committing encryption technique of Beaver and Haber [3], which assumes erasures. Jarecki and Lysyanskaya [19] criticised this erasure model and pointed out that the protocols presented in [6] and [10] are not secure in the concurrent setting, i.e., two instances of the same scheme can not be run at the same time. They solved this by introducing a "committed proof", i.e., a zero-knowledge proof where the statement that is being proved is not revealed until the end of the proof. To implement the secure channels without erasures they use an encryption scheme that is non-committing to the receiver. Abe and Fehr [1] later proposed an adaptively-secure (Feldman-based) DKG and applications with complete security proofs in the Universal Composability framework of Canetti [5]. They demonstrated that a discrete-log DKG protocol can be achieved without interactive zero-knowledge proofs. However, they still need a single inconsistent player and a secure message transmission functionality (private channels), which can be realised using a receiver non-committing transmission protocol based on [19].

As a consequence of private channels, each of the aforementioned DKG protocols has some kind of complaint procedure or dispute resolution mechanism. To get rid of these, several authors have proposed protocols that provide public verifiability. Stadler [29] was of the first to propose a publicly verifiable secret sharing (PVSS) protocol. In addition to the Feldman commitments, shares were broadcast in encrypted form and verified using a non-interactive proof of equality of (double) discrete logarithms. A more efficient protocol was presented by Fujisaki and Okamoto [11], which is secure under a modified RSA assumption. The first PVSS shown secure under the Decisional Diffie-Hellman (DDH) assumption

was given by Schoenmakers [26]. The shares are broadcast in encrypted form by hiding them in the exponent of each player's individual public key, which has a different base (another generator) than the Feldman commitments. The dealer then uses non-interactive proofs of discrete-log equality. Furthermore, correct behaviour of the players is verified by extending the secret reconstruction phase with additional proofs of correctness. Based on Schoenmakers' result Heidarvand and Villar [18] presented the first PVSS protocol where verifiability is obtained from bilinear pairings over elliptic curves and no proofs are needed. Unfortunately, the scheme cannot be used to set up a DKG because the shared secret is in the co-domain of the pairing. The first full DKG that does not require private channels was given by Fouque and Stern [9]. The buildings blocks for their construction are Paillier's cryptosystem and a new non-interactive zero-knowledge proof. To deal with rushing adversar it is simply assumed that communication is completely synchronous. For participants not present during the DKG the amount of information that needs to be stored, i.e., the subshares that need to be decrypted, is linear in the number of participants that are active in the DKG.

3 Basic Concepts

Before we describe our new protocols, we give an overview of basic concepts that will be used later on.

3.1 Threshold Cryptography

Threshold cryptography typically involves routines related to setting up the group, encryption and signatures. A private key is shared amongst n devices and only a subset of at least $t+1$ devices need to employ their shares to (implicitly) use this private key in a cryptosystem or signature scheme. We define the following set of routines (threshold routines are indicated with the prefix **T**):

Pre-setup.

- **Init:** Initialise the system parameters.
- **KeyGen:** Generate key material for a device.

Setup.

- **ConstructGroup:** Given a set of n devices and their public keys, create and share a key pair for the group with a subset of the devices.

Signatures.

- **T-Sign:** At least $t+1$ devices collaborate to generate a signature on a message that is verifiable under the group's public key.
- **Verify:** Using the group's public key a signature is verified.

Encryption.

- **Encrypt:** Encrypt a message under the group's public key.
- **T-Decrypt:** At least $t+1$ devices collaborate to decrypt a given ciphertext that was encrypted under the group's public key.

3.2 Communication and Adversarial Model

We assume that n devices $\{\mathcal{D}_i\}_{i=1..n}$, of which t can be faulty, communicate over a dedicated broadcast channel.[2] By dedicated we mean that if a device $\mathcal{D}_i$ broadcasts a message, then it is received by all other devices and recognised as coming from this device. There are no private channels, all communication goes over the broadcast channel. Communication is round-synchronous, protocols run in rounds and there is a time bound on each round.

A distinction is commonly made between static and adaptive adversaries. Static means that the adversary corrupts the devices before the protocol starts, whereas adaptive means that a device can become corrupt before or at any time during execution of a protocol. We assume a malicious computationally bounded static adversary who can corrupt up to t devices. The adversary has access to all information stored by the corrupted devices and can manipulate their behaviour during the execution of a protocol in any way. The round-synchronous communication implies that the adversary could be rushing, i.e., he can wait in each round to send messages on behalf of the corrupted devices until he has received the messages from all uncorrupted devices.

3.3 Security Definitions

In a secret sharing scheme a dealer splits a secret into pieces, called shares, and distributes them amongst several parties. In a threshold setting, the secret can be reconstructed from any subset of shares of a minimum size. An early solution was given by Shamir [27], who shared a secret x by choosing a random polynomial f of degree t such that $x = f(0)$ and each share is an evaluation of this polynomial, i.e., $x_i = f(i)$. Any point on a polynomial of degree t can be reconstructed by Lagrange interpolation through at least $t + 1$ points of this polynomial. To reconstruct the secret x, the shares are combined as $x = \sum \lambda_i x_i$, with λ_i the appropriate Lagrange multipliers.

Verifiable secret sharing (VSS) allows the receivers to verify that the dealer properly shared a secret. We briefly rephrase the requirements of a secure VSS ([23] and [15, Lemma 1]).

Definition 1 (Secure VSS). *A VSS protocol is secure if it satisfies the following conditions:*

1. *Correctness. If the dealer is not disqualified then any subset of $t + 1$ honest players can recover the unique secret.*
2. *Verifiability. Incorrect shares can be detected at reconstruction time by using the output of the protocol.*
3. *Secrecy. The view of a computationally bounded static adversary $\mathcal{A}$ is independent of the secret, or, the protocol is semantically secure against $\mathcal{A}$.*

The drawback of VSS is that a single party knows the secret. This can be solved by generating and sharing the key in a distributed way. The correctness and secrecy requirements for DKG were defined by Gennaro et al. [17].

[2] We abstract from the actual implementation of the dedicated broadcast channel.

Definition 2 (Secure DKG). *A DKG protocol is secure if it satisfies the following conditions:*
Correctness *is guaranteed if:*
(C1) All subsets of $t+1$ shares provided by honest players define the same private key.
(C2) All honest parties know the same public key corresponding to the unique private key as defined by (C1).
(C3) The private key (and thus also the public key) is uniformly distributed.
Secrecy *is guaranteed if an adversary can learn no information about the private key beyond what can be learnt from the public key. This requirement can be further enhanced with a simulation argument: for any adversary there should be a simulator that, given a public key, simulates a run of the protocol for which the output is indistinguishable of the adversary's view of a real run of the protocol that ended with the given public key.*

3.4 Pairings and Number-Theoretic Assumptions

We review some assumptions that are relevant for this paper and we refer the reader to [21] and [28] for more details.

Pairing Notation. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of order ℓ and let $\hat{e}$ be a non-degenerate bilinear pairing

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T \ .$$

A pairing is non-degenerate if for each element P in $\mathbb{G}_1$ there is a Q in $\mathbb{G}_2$ such that $\hat{e}(P, Q) \neq 1$ and vice versa for each element Q in $\mathbb{G}_2$. A pairing is bilinear if $\hat{e}(P + P', Q) = \hat{e}(P, Q)\hat{e}(P', Q)$, thus $\hat{e}(aP, Q) = \hat{e}(P, Q)^a$ with $a \in \mathbb{Z}_\ell$, and vice versa for elements in $\mathbb{G}_2$. We will use multiplicative notation for $\mathbb{G}_T$ and additive notation for $\mathbb{G}_1$ and $\mathbb{G}_2$.

Discrete Logarithm. Let P be a generator of $\mathbb{G}_1$ and let Y be a given arbitrary element of $\mathbb{G}_1$. The discrete logarithm (DL) problem in $\mathbb{G}_1$ is to find the unique integer $a \in \mathbb{Z}_\ell$ such that $Y = aP$. Similarly, the problem can be defined in $\mathbb{G}_2$ and $\mathbb{G}_T$. The DL assumption states that it is computationally hard to solve the DL problem.

Diffie-Hellman. Let P be a generator of $\mathbb{G}_1$ and let aP, bP be two given arbitrary elements of $\mathbb{G}_1$, with $a, b \in \mathbb{Z}_\ell$. The computational Diffie-Hellman (CDH) problem in $\mathbb{G}_1$ is to find abP. The tuple $\langle P, aP, bP, abP\rangle$ is called a Diffie-Hellman tuple. Given a third element $cP \in \mathbb{G}_1$, the decisional Diffie-Hellman (DDH) problem is to determine if $\langle P, aP, bP, cP\rangle$ is a valid Diffie-Hellman tuple or not. Obviously, if one can solve the DL problem then one can also solve the CDH problem. The opposite does not necessarily hold and, therefore, the CDH assumption is said to be a stronger assumption than the DL assumption. A divisional variant of the DDH problem [2], which is considered to be equivalent, is to determine if $\langle P, aP, cP, abP\rangle$ is a valid DH tuple or not, i.e., if $c = b$.

Co-Bilinear Diffie-Hellman (coBDH). For asymmetric pairings, i.e., $\mathbb{G}_1 \neq \mathbb{G}_2$, where there is no known efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ the following problem can be defined. The coBDH-2 problem is defined as given $P \in \mathbb{G}_1$ and $Q, aQ, bQ \in \mathbb{G}_2$, find $\hat{e}(P,Q)^{ab}$. We denote the decisional variant as coDBDH-2. A divisional variant of the coDBDH-2 problem is to determine whether $\langle P, Q, aQ, abQ, g^c \rangle$ is a valid coBDH-2 tuple.

Inversion Problems. Galbraith et al. [12] studied several inversion problems for pairings. They concluded that these problems are hard enough to rely upon. The most intuitive argument is that if one can solve a particular pairing inversion in polynomial time then one can also solve a related Diffie-Hellman problem in one of the domains or the co-domain.

3.5 Pre-setup

The pre-setup phase is straightforward and works as follows.

Init(1^k): The input is a security parameter k. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be finite cyclic groups of prime order ℓ with P, Q and $g = \hat{e}(P,Q)$ generators of the respective groups. It is assumed that there is no known efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$. Let P' and P'' be two other generators of $\mathbb{G}_1$ for which the discrete logarithm relative to the base P is unknown and let $g_1 = g$, $g_2 = \hat{e}(P', Q)$ and $g_3 = \hat{e}(P'', Q)$. The procedure outputs the description of the groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ and the pairing $(\hat{e})$ along with the public system parameters

$$PubPar = (P, P', P'', Q) \in \mathbb{G}_1^3 \times \mathbb{G}_2 \ .$$

KeyGen(*PubPar*,$\mathcal{D}_i$): For the given device $\mathcal{D}_i$ a random $s_i \in_R \mathbb{Z}_\ell^*$ is chosen as private key. The corresponding public key is $S_i = s_i Q$. The procedure outputs $\mathcal{D}_i$'s key pair

$$(s_i, S_i) \in \mathbb{Z}_\ell^* \times \mathbb{G}_2 \ .$$

Note that this procedure is executed only once in the lifetime of each $\mathcal{D}_i$ and that s_i is the only secret that has to be securely stored. Typically, this routine is executed during fabrication of the device. A public key can easily be computed for a different set of system parameters if this would be required.

4 Distributed Key Generation

In this section, we present our main result, which is a new distributed key generation (DKG) protocol. Recall from the introduction that we want to set up a threshold construction without the devices having to securely store their share. Instead, the shares will be stored in protected form. This idea is put forward in Sect. 4.1. Our DKG consists of two phases. First, a private key is jointly generated and shared through the parallel execution of a new publicly verifiable secret

sharing (PVSS) protocol. This PVSS protocol is described in Sect. 4.2. Second, the corresponding public key is extracted. Together, these two phases make up our new DKG protocol, which is presented in Sect. 4.3.

4.1 Protecting Shares

As mentioned in Sect. 3.5, each device $\mathcal{D}_i$ is initialised with its own key pair (s_i, S_i). If the group's private key was $x \in \mathbb{Z}_\ell$, then the device would receive a share $x_i \in \mathbb{Z}_\ell$ that has to be securely stored. Since the device does not provide secure storage it has to store its share in protected form. One option is to encrypt the share x_i. This has the disadvantages of involving a costly decryption operation and the fact that the share will at some point reside in the clear in the device's memory. Another option is to store the share as the product $x_i s_i$. The obvious disadvantage is that $t+1$ devices can collaborate to compute another device's private key s_i.

As we do not want a device's private key s_i ever to be revealed, we combine shares with the device's public key and store these as public correction factors $C_i = x_i S_i \in \mathbb{G}_2$. A similar idea was used in [26]. However, here we use bilinear pairings to achieve public verifiability and easy integration of our scheme in existing discrete-log cryptosystems and signature schemes, without ever having to reveal the shares (see Sect. 5). We define the group's private key as $xQ \in \mathbb{G}_2$ and $y = g^x = \hat{e}(P, xQ) \in \mathbb{G}_T$ as its public key.[3] As such, the share of a device is $x_i Q = s_i^{-1} C_i \in \mathbb{G}_2$. The construct group routine is formally defined as follows.

- **ConstructGroup(**$PubPar$**,**$\{\mathcal{D}_i, S_i\}$**,**t**):** A subset of the devices $\mathcal{D}_i$ generates the group's public key g^x and shares the private key xQ in the form of public correction factors $C_i = x_i S_i$ for all n devices. The procedure outputs the group's public key
$$y = g^x = \hat{e}(P, xQ) \in \mathbb{G}_T$$
and the public correction factors which are added to the public parameters
$$PubPar = (P, P', P'', Q, y, \{C_i\}_{i=1,\ldots,n}) \in \mathbb{G}_1^3 \times \mathbb{G}_2 \times \mathbb{G}_T \times \mathbb{G}_2^n \ .$$

4.2 Publicly Verifiable Secret Sharing

The main building block to construct our DKG protocol is a new PVSS protocol. In this protocol, a dealer generates shares of a secret and distributes them in protected form. Any party observing the output of the protocol can verify that the dealer behaved correctly. Basically, the protocol goes as follows.

The dealer chooses uniformly at random $x \in_R \mathbb{Z}_\ell$. The actual secret that is shared at the end of the protocol is xQ. Similar to Pedersen's VSS scheme [23], the dealer chooses two random polynomials f and f' of degree t, sets the constant

[3] We note that we could use notation X and X_i for the private key and its shares. However, to compute the correction factor C_i, elements of $\mathbb{Z}_\ell$ will be combined with S_i, but by definition, private key material is in $\mathbb{G}_2$.

term of polynomial f to x and broadcasts pairwise commitments $A_k \in \mathbb{G}_1$ to the coefficients of the polynomials. Evaluations of both polynomials are combined with the public keys of the devices and broadcast in protected form. Each device verifies that all broadcast shares are correct by applying the pairing to check them against the commitments. The details of the protocol are given in Fig. 1.

The dealer shares the secret xQ, for which he chooses $x \in_R \mathbb{Z}_\ell$:

1. The dealer constructs two polynomials $f(z)$ and $f'(z)$ of degree t by choosing random coefficients $c_k, c'_k \in_R \mathbb{Z}_\ell^*$ for $k = 0 \ldots t$, except for c_0, which is $c_0 = x$:

$$f(z) = c_0 + c_1 z + \ldots + c_t z^t \quad , \quad f'(z) = c'_0 + c'_1 z + \ldots + c'_t z^t \ .$$

   The dealer broadcasts commitments

$$A_k = c_k P + c'_k P' \quad , \quad k = 0 \ldots t \ .$$

2. For each device $\mathcal{D}_i$, the dealer computes and broadcasts

$$x_i S_i \ , \ x'_i S_i \quad \text{with} \quad x_i = f(i) \ , \ x'_i = f'(i) \quad , \quad i = 1 \ldots n \ .$$

3. Each device verifies the broadcast shares for all $\mathcal{D}_i$ by checking that

$$\hat{e}(P, x_i S_i) \cdot \hat{e}(P', x'_i S_i) = \prod_{k=0}^{t} \hat{e}(A_k, S_i)^{i^k} \ . \tag{1}$$

   If any of these checks fails, the dealer is disqualified.

Fig. 1. Publicly verifiable secret sharing.

Private channels are avoided because the shares $x_i Q$ are broadcast in protected form $x_i S_i$. Each device could recover its share by using its private key. However, the shares are never needed in unprotected form. The protected form allows for public verifiability, since for any device $\mathcal{D}_i$ the correctness of $x_i S_i$ and $x'_i S_i$ can be verified by pairing the commitments with $\mathcal{D}_i$'s public key S_i. The dealer is disqualified, if for any $\mathcal{D}_i$ this verification fails. As a consequence, there is no need for a cumbersome complaint procedure. Moreover, not all devices need to be present during the execution of the protocol because the shares were already broadcast in the form in which they will be stored and used.

In the next theorem we will demonstrate that our new PVSS protocol satisfies the requirements of secure VSS protocol as given by Definition 1.

Theorem 1. *Our new PVSS protocol is a secure VSS protocol (Definition 1) under the divisional variant of the DDH assumption in* $\mathbb{G}_2$.

Proof (Correctness). It follows directly from Pedersen's result [23] that each subset of $t+1$ devices can reconstruct the coefficients $c_k Q, c'_k Q$ of the polynomials

$F(z) = f(z)Q$ and $F'(z) = f'(z)Q$ from their shares. If the dealer is not disqualified then (1) holds for all devices and the coefficients will successfully be verified against the commitments A_k. Hence, it can be verified that all shares are on the same (respective) polynomial and each subset of $t+1$ devices can compute the same secret $xQ = F(0)$. □

Proof (Verifiability). During reconstruction $\mathcal{D}_i$ provides x_iQ and x'_iQ, and it can be verified that $\hat{e}(P, x_iQ) \cdot \hat{e}(P', x'_iQ) = \prod_{k=0}^{t} \hat{e}(A_k, Q)^{i^k}$. □

Proof (Secrecy). Consider a worst-case static adversary $\mathcal{A}$, i.e., an adversary that corrupts t devices before the protocol starts. The protocol is semantically secure against $\mathcal{A}$, if $\mathcal{A}$ chooses two values $x_0Q, x_1Q \in \mathbb{G}_2$ and cannot determine which of these two was shared with negligible advantage over random guessing, given the output of a run of the protocol that shared either the secret x_0Q or x_1Q. We prove the semantic security by showing that no such adversary exists.

If there exists an $\mathcal{A}$ that has a non-negligible advantage in attacking the semantic security of our protocol, we can build a simulator SIM that uses $\mathcal{A}$ to solve an instance of the divisional DDH problem in $\mathbb{G}_2$ (see Sect. 3.4). Since, this is assumed to be a hard problem we conclude that no such adversary can exist.

We now describe this simulator. A tuple $\langle Q, aQ, cQ, abQ \rangle$ is given to SIM who has to decide if this is a valid DH tuple, i.e., if $cQ = bQ$.

1. The simulator SIM does the pre-setup. He chooses the system parameters *PubPar*, which contain P and $P' = \eta P$, with η known to SIM. He constructs a set of devices $\mathcal{D}_i$, of which one will be the designated device, denoted as $\mathcal{D}_d$. For each $\mathcal{D}_i \neq \mathcal{D}_d$, SIM generates a random key pair. The public key of $\mathcal{D}_d$ is set to $S_d = cQ$.
2. The adversary $\mathcal{A}$ receives *PubPar* and the set of devices along with their public keys. He announces the subset of corrupted devices, which will be denoted by $\mathcal{D}_j$ for $j = 1 \ldots t$.
3. The simulator SIM gives the private keys s_j of the corrupted devices to $\mathcal{A}$. Device $\mathcal{D}_d$ is corrupted with a worst-case probability of roughly $1/2$, in which case the simulation fails.
4. $\mathcal{A}$ outputs two values x_0Q and x_1Q, of which one has to be shared.
5. Without loss of generality, we assume SIM chooses x_0Q. The output of the VSS protocol is generated as follows.
   - SIM chooses k random $z_k \in_R \mathbb{Z}_\ell^*$ and broadcasts commitments $A_k = z_kP$.
   - SIM constructs a random polynomial $F(z)$ of degree t subject to $F(0) = x_0Q$ and $F(d) = aQ$. For (1) to hold, future shares x_iQ and x'_iQ will have to satisfy

$$\alpha_iQ = x_iQ + \eta x'_iQ \quad \text{with} \quad \alpha_i = \sum_{k=0}^{t} z_k i^k \ . \tag{2}$$

   SIM evaluates the polynomial $F(z)$ and sets the shares $x_jQ = F(j)$ for each corrupted $\mathcal{D}_j$. For the non-corrupted $\mathcal{D}_i \neq \mathcal{D}_d$, SIM chooses random shares $x_iQ \in_R \mathbb{G}_2$. For $i \neq j$, the shares on the second polynomial x'_iQ and x'_jQ are determined by (2).

- With the private keys s_i and s_j, SIM computes the protected shares x_iS_i, x'_iS_i and x_jS_j, x'_jS_j.
- For $\mathcal{D}_d$, SIM sets $x_dS_d = abQ$ and $x'_dS_d = \eta^{-1}(\alpha_dS_d - abQ)$.
- All protected shares are broadcast by SIM.

6. The adversary outputs a guess to which of the secrets was shared. If $\mathcal{A}$ has a non-negligible advantage in determining which secret was shared then SIM concludes that $\langle Q, aQ, cQ, abQ\rangle$ must be a valid DH tuple.

The view of $\mathcal{A}$ consists of the commitments A_k, all public keys, the private keys of the corrupted devices, all protected shares and the shares of the corrupted devices. The adversary $\mathcal{A}$ can only gain an advantage in guessing which key was shared from values, other than his own shares, which were not chosen at random. This leaves him with only his shares x_jQ and the values x_dS_d and S_d. The adversary's problem of deciding which secret was shared is equivalent to deciding whether $x_dQ = x_0Q - \sum \lambda_j x_j Q$ or $x_dQ = x_1Q - \sum \lambda_j x_j Q$. Because we assume SIM chose x_0Q, $\mathcal{A}$ has to decide whether $\langle Q, x_0Q - \sum \lambda_j x_j Q, S_d, x_dS_d\rangle$ is a valid DH tuple or not. □

We note that given the specific form in which the shares are broadcast, our PVSS protocol cannot be proved secure against an adaptive adversary by means of a simulation argument, which does not imply that it is insecure. Indeed, it was already suggested in [6] and [10] that to maintain private transmission of shares some form of non-committing encryption should be used. We insist on storing shares as x_iS_i in order to maintain the nice properties of this form, which allow integrating our construction in other threshold applications, as shown in Sect. 5.

A somewhat related PVSS scheme was presented by Heidarvand and Villar [18].[4] Our PVSS scheme differs from theirs by putting the secret in $\mathbb{G}_2$, instead of $\mathbb{G}_T$, and thus allowing it to be a building block for DKG and discrete-log constructions. Moreover, our protocol is semantically secure while the scheme in [18] is only proved to be secure under a weaker security definition, because the adversary is not allowed to choose the secrets that he has to distinguish.

4.3 Distributed Key Generation.

We now establish a new DKG protocol that outputs protected shares and is publicly verifiable. Inspired by [15] and [6] the protocol consists of two phases. In the first phase, the group's private key is generated in a distributive manner and shared through a joint PVSS. In the second phase, the group's public key is computed. This phase follows to a large extent the result of Canetti et al. [6]. The protocols proceeds as follows.

Each participating device runs an instance of our new PVSS protocol. It chooses a secret $c_{i,0} \in_R \mathbb{Z}_\ell$ and broadcasts shares of that secret in protected form. These will be denoted as protected subshares. Each device, acting as a dealer, that is not disqualified is added to a set of qualified devices, denoted as

[4] Note that we use an asymmetric pairing which is more standard (e.g., see [13]) than the symmetric form used in [18].

QUAL. The group's private key, although never computed explicitly, is defined as $xQ = \sum_{i \in \text{QUAL}} c_{i,0} Q$. A device's protected share $x_i S_i$ is computed as the sum of the protected subshares that were received from the devices in QUAL.

To recover the group's public key $y = g^x$, the qualified devices will expose g^{x_i} from which y can easily be computed through Lagrange interpolation.[5] Each device will prove in zero-knowledge that the exponent of g^{x_i} matches the share $x_i Q$ hidden in $x_i S_i$, without revealing it. These interactive zero-knowledge proofs require uniformly distributed challenges, which can be the same for all devices.

A uniformly distributed challenge is generated through another run of our joint PVSS. All devices receive protected shares $d_i S_i$. After open reconstruction we have a uniformly distributed element $dQ \in \mathbb{G}_2$. However, the challenge needs to be some element $\tilde{d} \in \mathbb{Z}_\ell$. This implies a bijective (not necessarily homomorphic) mapping $\psi : \mathbb{G}_2 \to \mathbb{Z}_\ell$. An example of such a mapping is to take the x-coordinate of dQ modulo ℓ, as is used in ECDSA signatures. Several issues have been reported with this mapping and alternatives, e.g., taking the sum of the x and the y-coordinates modulo ℓ [20], have been proposed. We refer the reader to [4] for a more in-depth treatment of this subject.

The details of the protocols are given in Fig. 2. Note that on the one hand, at least $t + 1$ honest devices are required for the protocol to end successfully, hence we require $n > 2t$. On the other hand, since we require no explicit private channels, only a minimum of $t + 1$ honest devices must participate in the DKG.

We now prove that our new DKG protocol is a secure DKG protocol according to the requirements specified in Definition 2.

Theorem 2. *Our new DKG protocol is a secure DKG protocol (Definition 2) under the divisional variant of the coDBDH-2 assumption.*

Proof (Correctness). All honest devices construct the same set of qualified devices QUAL since this is determined by public broadcast information.

– (C1) Each $\mathcal{D}_i$ that is in QUAL at the end of phase 1 has successfully shared $c_{i,0}Q$ through a run of our PVSS protocol. Any set of $t+1$ honest devices $\mathcal{D}_i$ that combine correct shares $x_j Q$ can reconstruct the same secret xQ since

$$\begin{aligned} xQ &= \sum_{i \in \text{QUAL}} c_{i,0} Q = \sum_{i \in \text{QUAL}} \left(\sum_j \lambda_j x_{ij} Q \right) \\ &= \sum_j \lambda_j \sum_{i \in \text{QUAL}} x_{ij} Q = \sum_j \lambda_j x_j Q \ . \end{aligned}$$

In the key extraction phase of our protocol at least $t + 1$ values g^{x_i} have been exposed and thus using interpolation g^{x_j} can be computed for any $\mathcal{D}_j$. This allows to tell apart correct shares from incorrect ones.

[5] As opposed to [6], we do not expose $g^{c_{i,0}}$, which avoids the costly reconstruction of the $g^{c_{j,0}}$ of the qualified devices that no longer participate in the second phase.

1. All participating devices $\mathcal{D}_i$ run the PVSS protocol simultaneously, the protected subshares are only broadcast after receiving all commitments from all $\mathcal{D}_i$.
   (a) Each $\mathcal{D}_i$ constructs two polynomials $f_i(z)$ and $f'_i(z)$ of degree t by choosing random coefficients $c_{i,k}, c'_{i,k} \in_R \mathbb{Z}_\ell^*$ for $k = 0 \ldots t$:

   $$f_i(z) = c_{i,0} + c_{i,1}z + \ldots + c_{i,t}z^t \quad , \quad f'_i(z) = c'_{i,0} + c'_{i,1}z + \ldots + c'_{i,t}z^t \, ,$$

   and broadcasts commitments

   $$A_{i,k} = c_{i,k}P + c'_{i,k}P' \quad , \quad k = 0 \ldots t \ .$$

   (b) For each device $\mathcal{D}_j$, each $\mathcal{D}_i$ computes and broadcasts

   $$x_{ij}S_j \ , \ x'_{ij}S_j \quad \text{with} \quad x_{ij} = f_i(j) \ , \ x'_{ij} = f'_i(j) \ .$$

   (c) Each device verifies the broadcast shares for all $\mathcal{D}_i$ by checking that

   $$\hat{e}(P, x_{ij}S_j) \cdot \hat{e}(P', x'_{ij}S_j) = \prod_{k=0}^{t} \hat{e}(A_{i,k}, S_j)^{j^k} \ .$$

   Each $\mathcal{D}_i$ that is not disqualified as a dealer is added to the list of qualified devices, denoted by QUAL. The group's private key is defined as $xQ = \sum_{i \in \mathrm{QUAL}} c_{i,0}Q$. For each $\mathcal{D}_i$ its protected share is computed as

   $$C_i = x_i S_i = \sum_{j \in \mathrm{QUAL}} x_{ji} S_i \ .$$

2. The qualified devices expose g^{x_i} to compute the public key $y = g^x$.
   (a) Each $\mathcal{D}_i$ in QUAL broadcasts g^{x_i} and $s_i P''$. It is easily verified that $\hat{e}(s_i P'', Q) = \hat{e}(P'', S_i)$. In addition, $\mathcal{D}_i$ chooses a random $r_i \in_R \mathbb{Z}_\ell^*$ and broadcasts commitments g^{r_i} and $r_i S_i$.
   (b) Generation of the uniform challenge, needed in the zero-knowledge proof.
   - Devices in QUAL run a Joint PVSS and obtain protected shares $d_i S_i$ and $d'_i S_i$, which are broadcast and verified. We denote the commitments of this Joint PVSS as $B_{i,k}$.
   - Open reconstruction of dQ. Devices in QUAL broadcast $d_i Q$ and $d'_i Q$. These are verified by checking that

   $$\hat{e}(P, d_i Q) \cdot \hat{e}(P', d'_i Q) = \prod_{k=0}^{t} \hat{e}(B_k, Q)^{j^k} \qquad \text{for} \qquad B_k = \sum_{i \in QUAL} B_{i,k} \ .$$

   - Let $\tilde{d} = \psi(dQ)$, where ψ is a bijective map from $\mathbb{G}_2$ to $\mathbb{Z}_\ell$.

   (c) Each $\mathcal{D}_i$ broadcasts $Z_i = s_i^{-1}(r_i S_i + \tilde{d} C_i) = (r_i + \tilde{d} x_i)Q$. Any device can verify that

   $$\hat{e}(P, Z_i) = g^{r_i}(g^{x_i})^{\tilde{d}} \quad \text{and} \quad \hat{e}(s_i P'', Z_i) = \hat{e}(P'', r_i S_i) \cdot \hat{e}(P'', C_i)^{\tilde{d}} \ .$$

   (d) Public key y is computed from $t+1$ correctly verified g^{x_i} as $y = \prod g^{x_i \lambda_i}$.

Fig. 2. Publicly verifiable DKG with protected shares.

- (C2) This follows immediately from the key extraction phase and the relation between the $c_{i,0}Q$ and the shares x_iQ given for the previous property (C1).
- (C3) The private key is defined as $xQ = \sum_{i \in QUAL} c_{i,0}Q$ and each $c_{i,0}Q$ was shared through an instance of our PVSS. Since we proved that a static adversary cannot learn any information about the shared secret, the private key is uniformly distributed as long as one non-corrupted device successfully contributed to the sum that defines xQ. □

Uniformity. Our protocol withstands the attack of a rushing adversary that can influence the distribution of the group's key as described by Gennaro et al. [17]. In this attack an adversary is able to compute a deterministic function of the private key from the broadcasts, before sending out his contributions. He can influence the set of qualified devices by choosing whether or not to send out proper contributions. This allows influencing the outcome of the deterministic function and thus the distribution of the private key. In our protocol, no such function can be computed before the second phase. But, because the private key and thus also the correction factors are fixed after the first phase and determined by QUAL, the adversary can no longer influence the group's key. As long as $t+1$ honest devices participate, the public key can be recovered in the second phase.

Proof (Secrecy). We describe a simulator SIM that, given a public key y, simulates a run of the protocol and produces an output that is indistinguishable from the adversary's view of a real run of the protocol that ended with the given public key. We assume that SIM knows $\eta \in \mathbb{Z}_\ell^*$ for which $P' = \eta P$.

- The first phase of the DKG is run as in the real protocol. Since SIM knows the private keys s_i of at least $t+1$ non-corrupted devices, he knows at least $t+1$ shares $x_iQ = s_i^{-1}C_i$. By interpolation of these shares, SIM learns the shares of the corrupted devices. This allows SIM to compute g^{x_i} for all devices.
- In the second phase of the DKG protocol SIM sets $g^{x_i^*}$ for the non-corrupted $\mathcal{D}_i$, such that the public key will be y. The $g^{x_i^*}$ for the non-corrupted $\mathcal{D}_i$ are calculated by interpolation of the g^{x_j} of the corrupted $\mathcal{D}_j$ and $y = g^x$. For the zero knowledge proof to hold, SIM chooses a random $d^* \in_R \mathbb{Z}_\ell^*$ and forces the outcome of the open reconstruction of the challenge to d^*Q. For each non-corrupted $\mathcal{D}_i$, SIM computes the commitments $\beta_i^* = g^{z_i}(g^{x_i^*})^{-\tilde{d}}$ and $B_i^* = z_iS_i - \tilde{d}x_iS_i$, for random $z_i \in_R \mathbb{Z}_\ell^*$ and $\tilde{d} = \psi(d^*Q)$.
  (a) SIM broadcasts $\langle g^{x_i^*}, s_iP'', \beta_i^*, B_i^* \rangle$ for each non-corrupted $\mathcal{D}_i$.
  (b) All devices run the Joint PVSS and hold shares d_iQ and d'_iQ. SIM forces the outcome of the open reconstruction of the challenge to d^*Q.
    - SIM computes the d_jQ from the corrupted devices by interpolation of $t+1$ shares d_iQ of the non-corrupted devices.
    - SIM sets the d_i^*Q for the non-corrupted devices by interpolation of the d_jQ of the corrupted devices and d^*Q.
    - By knowing η, SIM will compute $d_i'^*Q$ such that $d_iQ + \eta d'_iQ = d_i^*Q + \eta d_i'^*Q$. As such, the broadcast shares d_i^*Q, $d_i'^*Q$, will verify against the commitments.

(c) All $Z_i^* = z_i Q$ are broadcast and correctly verified.
(d) At the end of the protocol the public key is computed as the given y.

To prevent an adversary from being able to distinguish between a real run of the protocol and a simulation, the output distribution must be identical. The first phase, i.e., the Joint PVSS, is identical in both cases. The data that are output in the second phase and that have a potentially different distribution in a real run and simulation are given in the following table. We show that all data in this table have a uniform distribution.

REAL	SIM
1. g^{x_i}	$g^{x_i^*}$
2. $g^{r_i}, r_i S_i$	β_i^*, B_i^*
3. $d_i Q, d_i' Q$	$d_i^* Q, d_i'^* Q$
4. Z_i	Z_i^*

1. The values x_i are evaluations of a polynomial of degree t with uniformly random coefficients. The values x_i^* are evaluations of a polynomial that goes through t evaluations of the first polynomial, namely the x_j of the corrupted participants, and through the discrete logarithm of y. Since the protocol is assumed to generate a uniformly random key, the new polynomial's distribution is indistinguishable from the distribution of the first.
2. The value r_i was chosen uniformly at random. In the simulation $\beta_i^* = g^{z_i}(g^{x_i^*})^{-\tilde{d}}$ and $B_i^* = z_i S_i - \tilde{d} x_i S_i$. The value z_i is uniformly random and $\tilde{d} = \psi(d^* Q)$ is derived from the uniformly random d^*.
3. Since the following relation holds, $d_i Q + \eta d_i' Q = d_i^* Q + \eta d_i'^* Q$, it suffices to show that both $d_i Q$ and $d_i^* Q$ have identical distributions. Because d was chosen uniformly at random, the same reasoning as for the g^{x_i} holds.
4. We have that $Z_i = r_i Q + \eta x_i Q$ and $Z_i^* = z_i Q$. The values r_i, d and z_i were chosen uniformly at random.

We notice that even though the modified $g^{x_i^*}$ have the right output distribution, it is important to note that by broadcasting the modified $g^{x_i^*}$ we introduce a new assumption. Namely that an adversary cannot distinguish between $\langle P, Q, x_i s_i Q, s_i Q, g^{x_i} \rangle$ and $\langle P, Q, x_i s_i Q, s_i Q, g^{x_i^*} \rangle$. This is the divisional variant of the coDBDH-2 assumption, as defined in Sect. 3.4. An adversary cannot distinguish $\langle Q, S_i, d_i Q, d_i S_i \rangle$ from $\langle Q, S_i, d_i^* Q, d_i S_i \rangle$. This is the divisional variant of the DDH assumption, which is a weaker assumption than the coDBDH-2 assumption, meaning that if one could not solve the coDBDH-2 problem, one can also not solve the DDH problem. Knowledge of P allows to calculate g^{d_i} and $g^{d_i^*}$ and to transform this to the divisional variant of the coDBDH-2 assumption. □

5 Threshold Applications

In this section our construction is used to turn discrete-log schemes into threshold variants with protected shares. It is not our intention to give a rigorous proof of

security of these variants. We rather want to demonstrate the ease with which our construction fits into existing schemes. We do this for the ElGamal [14] and the Cramer-Shoup [7] cryptosystems, where we show how pairings allow implicit use of the shares, i.e., without having to reveal them explicitly, and the Schnorr [25] signature scheme.

5.1 ElGamal

Basic Scheme. We define the ElGamal [14] scheme in $\mathbb{G}_T$ with some minor modifications; the randomness is moved from $\mathbb{G}_T$ to $\mathbb{G}_1$ and the private key is an element of $\mathbb{G}_2$ instead of $\mathbb{Z}_\ell^*$, i.e., $xQ \in \mathbb{G}_2$ for some $x \in_R \mathbb{Z}_\ell^*$. Let $y = \hat{e}(P,Q)^x$ be the corresponding public key. Encryption and decryption are defined as follows.

- **Encrypt(***PubPar***,***y***,***m***):** To encrypt a message $m \in \mathbb{G}_T$ under the public key y, choose a random $k \in_R \mathbb{Z}_\ell^*$ and output the ciphertext

$$(R, e) = (kP, my^k) \in \mathbb{G}_1 \times \mathbb{G}_T \ .$$

- **Decrypt(***PubPar***,***xQ***,**(R,e)**):** To decrypt the given ciphertext (R, e) output the plaintext

$$m = \frac{e}{\hat{e}(R, xQ)} \in \mathbb{G}_T \ .$$

Threshold Variant. Encryption in the threshold variant is the same as in the basic scheme. To decrypt a given ciphertext we have to combine the randomness kP with $t+1$ shares x_iQ, which are stored as x_iS_i. If the shares were stored as $g^{x_i s_i}$, it would have been impossible to combine them with the randomiser or the ElGamal encryption and for each device $\mathcal{D}_i$ the ciphertext would contain something like $g^{x_i s_i k}$. By taking advantage of the bilinearity of the pairing, the size of the ciphertext remains constant. Note that $\mathcal{D}_i$ never has to reveal his share explicitly; his private key is combined with the randomness and then paired with the correction factor. The cost of providing a partial decryption is minimal, namely one elliptic-curve point multiplication. In this way we can use small devices as partial decryption oracles. Decryption goes as follows.

- **T-Decrypt(***PubPar***,**$\{\mathcal{D}_i, S_i\}$**,**(R,e)**):** To decrypt the ciphertext (R, e) each device $\mathcal{D}_i$ provides a partial decryption

$$D_i = s_i^{-1} R = s_i^{-1} kP \in \mathbb{G}_1 \ .$$

  The combining device receives the D_i and verifies that $\hat{e}(D_i, S_i) = \hat{e}(R, Q)$. He then combines $t+1$ contributions to output the plaintext

$$m = \frac{e}{d} \quad \text{with} \quad d = \prod \hat{e}(D_i, C_i)^{\lambda_i} \ .$$

5.2 Cramer-Shoup

Basic Scheme. Cramer and Shoup [7] presented an ElGamal based cryptosystem in the standard model that provides ciphertext indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2). We define their scheme in $\mathbb{G}_T$ with the same modifications as in the ElGamal scheme; the first two (random) elements in the ciphertext are moved from $\mathbb{G}_T$ to $\mathbb{G}_1$ and the private key is a tuple from $\mathbb{G}_2^5$ instead of $(\mathbb{Z}_\ell^*)^5$. Let $H : \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_T \to \mathbb{Z}_\ell$ be an element of a family of universal one-way hash functions. The private key is

$$privK = (x_1 Q, x_2 Q, y_1 Q, y_2 Q, zQ) \in_R \mathbb{G}_2^5$$

and the public key is

$$pubK = (c, d, h) = (g_1^{x_1} g_2^{x_2}, g_1^{y_1} g_2^{y_2}, g_1^z) \in \mathbb{G}_T^3 \ .$$

Encryption and decryption are defined as follows.

- **Encrypt(***PubPar***,***pubK***,***m***):** To encrypt a message $m \in \mathbb{G}_T$ under *pubK*, choose a random $k \in_R \mathbb{Z}_\ell$ and output the ciphertext

  $$(U_1, U_2, e, v) = (kP, kP', mh^k, c^k d^{k\alpha}) \in \mathbb{G}_1^2 \times \mathbb{G}_T^2 \quad \text{with} \quad \alpha = H(U_1, U_2, e) \ .$$

- **Decrypt(***PubPar***,***privK***,**(U_1, U_2, e, v)**):** To decrypt ciphertext (U_1, U_2, e, v), first compute $\alpha = H(U_1, U_2, e)$ and validate the ciphertext by testing if

  $$\hat{e}(U_1, x_1 Q + y_1 \alpha Q) \cdot \hat{e}(U_2, x_2 Q + y_2 \alpha Q) = v \ .$$

  If the test fails, the ciphertext is rejected, otherwise output the plaintext

  $$m = \frac{e}{\hat{e}(U_1, zQ)} \in \mathbb{G}_T \ .$$

Threshold Variant. It is clear that the Cramer-Shoup public key is not immediately established from running five instances of our DKG protocol. The decomposition of $c = g_1^{x_1} g_2^{x_2}$ and $d = g_1^{y_1} g_2^{y_2}$ should not be known. We can solve this problem by introducing a third polynomial $f''(z)$. Each device receives three instead of two shares. The public key is extracted by revealing the third share and by proving the discrete log equality of $g_3^{x_i''}$ and $x_i'' S_i$. The DKG is thereby reduced to two runs of the variant and one run of the basic DKG protocol. This results in five protected shares $C_i^{x_1}, C_i^{x_2}, C_i^{y_1}, C_i^{y_2}$ and C_i^z for each device.

Encryption is the same as in the basic scheme. The decryption routine, which applies the same ideas as in the threshold ElGamal scheme goes as follows. Note that the cost of providing a partial decryption is minimal, namely two elliptic-curve point multiplications.

- **T-Decrypt(***PubPar***,**$\{\mathcal{D}_i, S_i\}$**,**(U_1, U_2, e, v)**):** To decrypt the given ciphertext (U_1, U_2, e, v) each device $\mathcal{D}_i$ provides $D_i = s_i^{-1} U_1$ and $D_i' = s_i^{-1} U_2$. The

combining device verifies that $\hat{e}(D_i, S_i) = \hat{e}(U_1, Q)$ and $\hat{e}(D'_i, S_i) = \hat{e}(U_2, Q)$. He then computes

$$v_i = \hat{e}(D_i, C_i^{x_1} + \alpha C_i^{y_1}) \cdot \hat{e}(D'_i, C_i^{x_2} + \alpha C_i^{y_2}) \ .$$

and combines $t+1$ values v_i to validate the ciphertext by testing that $v = \prod v_i^{\lambda_i}$. If validation fails, the ciphertext is rejected. The combining device combines $t+1$ contributions to output the plaintext

$$m = \frac{e}{d} \quad \text{with} \quad d = \prod \hat{e}(D_i, C_i^z)^{\lambda_i} \ .$$

5.3 Schnorr Signatures

The Schnorr signature scheme [25] is an example of a scheme that provides existential unforgeability under an adaptive chosen-message attack in the random oracle model [24] and has been used many times to create a threshold signature scheme, e.g., in [17,1]. We will define the signature scheme in $\mathbb{G}_T$ and then extend it to a threshold variant.

Basic Scheme. Let $H' : \{0,1\}^* \times \mathbb{G}_T \to \mathbb{Z}_\ell$ be a cryptographic hash function. Let the private key be $xQ \in \mathbb{G}_2$ for $x \in_R \mathbb{Z}_\ell^*$ and $y = g^x \in \mathbb{G}_T$ the public key.

- **Sign(***PubPar*,xQ,m**):** To sign a message $m \in \{0,1\}^*$ with the private key xQ choose a random $k \in_R \mathbb{Z}_\ell$, compute $r = \hat{e}(P, kQ)$ and $c = H'(m, r)$, and output the signature

$$(c, \sigma) = (H'(m, r), kQ + c\,xQ) \in \mathbb{Z}_\ell \times \mathbb{G}_2 \ .$$

- **Verify(***PubPar*,y,(c,σ),m**):** To verify the signature (c, σ) on a message m compute $\tilde{r} = \hat{e}(P, \sigma) y^{-c}$ and verify equality of $c = H'(m, \tilde{r})$.

Threshold Variant. The basic scheme naturally extends to a threshold variant. As opposed to the encryption schemes, the bilinearity of the pairing is not really needed. However, the signing devices need to share some randomness and will, therefore, run the DKG protocol of Sect. 4.3. Signature verification is the same as in the basic scheme. Signing goes as follows.

- **T-Sign(***PubPar*,$\{\mathcal{D}_i\}$,m**):** To sign a message $m \in \{0,1\}^*$ with the group's private key the devices $\mathcal{D}_i$ will run an instance of the DKG protocol of Sect. 4.3. Each device then holds a share $k_i S_i$ in protected form of $kQ \in \mathbb{G}_2$. Because the value $r = \hat{e}(P, kQ) \in \mathbb{G}_T$ is publicly computed at the end of the protocol, each device can compute $c = H'(m, r)$ and $\sigma_i = s_i^{-1}(k_i S_i + cC_i) = (k_i + c\,x_i)Q$, which is sent to the combining device. Note that these partial signatures can be verified since the output of the DKG protocols contained g^{k_i} and g^{x_i}. Values that were not in the output can be computed through interpolation. The combining device computes the signing equation $\sigma = \sum \sigma_i \lambda_i$ and outputs the signature

$$(c, \sigma) = (H'(m, r), kQ + c\,xQ) \in \mathbb{Z}_\ell \times \mathbb{G}_2 \ .$$

6 Conclusion

In this paper, we have shown how to increase resilience in threshold cryptography by including small devices with limited or no secure storage capabilities. Assuming these devices have some support for public-key functionality, shares can be stored in protected form. By using bilinear pairings, this particular form yields some advantages. The most important feature is public verifiability, which makes explicit private channels and cumbersome complaint procedures obsolete. Moreover, not all devices need to be present during group setup, which is performed by the DKG protocol. We have demonstrated how to adopt the protected shares in existing discrete-log based cryptosystems and signature schemes. Because shares are never needed in unprotected form, small devices can be used as decryption oracles at a minimal cost.

Acknowledgments. The authors would like to thank Frederik Vercauteren, Alfredo Rial Duran and Markulf Kohlweiss for the fruitful discussions. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the IST programmes under contract ICT-2007-216676 ECRYPT II. Roel Peeters is funded by a research grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

References

1. Abe, M., Fehr, S.: Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 317–334. Springer-Verlag (2004)
2. Bao, F., Deng, R.H., Zhu, H.: Variations of Diffie-Hellman problem. In: Qing, S., Gollmann, D., Zhou, J. (eds.) ICICS 2003. LNCS, vol. 2836, pp. 301–312. Springer-Verlag (2003)
3. Beaver, D., Haber, S.: Cryptographic protocols provably secure against dynamic adversaries. In: Rueppel, R.A. (ed.) EUROCRYPT '92. LNCS, vol. 658, pp. 307–323. Springer-Verlag (1992)
4. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Designs, Codes and Cryptography 35(1), 119–152 (2005)
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001. pp. 136–145. IEEE Computer Society (2001)
6. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Wiener, M.J. (ed.) CRYPTO '99. LNCS, vol. 1666, pp. 98–115. Springer-Verlag (1999)
7. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO '98. LNCS, vol. 1462, pp. 13–25. Springer-Verlag (1998)
8. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: FOCS 1987. pp. 427–437. IEEE Computer Society (1987)

9. Fouque, P.A., Stern, J.: One round threshold discrete-log key generation without private channels. In: PKC 2001. LNCS, vol. 1992, pp. 300–316. Springer-Verlag (2001)
10. Frankel, Y., MacKenzie, P.D., Yung, M.: Adaptively-secure distributed public-key systems. In: Nesetril, J. (ed.) ESA '99. LNCS, vol. 1643, pp. 4–27. Springer-Verlag (1999)
11. Fujisaki, E., Okamoto, T.: A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In: Nyberg, K. (ed.) EUROCRYPT '98. LNCS, vol. 1403, pp. 32–46. Springer-Verlag (1998)
12. Galbraith, S., Hess, F., Vercauteren, F.: Aspects of pairing inversion. IEEE Transactions on Information Theory 54(12), 5719–5728 (2008)
13. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), http://eprint.iacr.org/
14. Gamal, T.E.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO '84. LNCS, vol. 196, pp. 10–18. Springer-Verlag (1985)
15. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT '99. LNCS, vol. 1592, pp. 295–310. Springer-Verlag (1999)
16. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure applications of Pedersen's distributed key generation protocol. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 373–390. Springer-Verlag (2003)
17. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. of Cryptology 20(1), 51–83 (2007)
18. Heidarvand, S., Villar, J.L.: Public verifiability from pairings in secret sharing schemes. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 294–308. Springer-Verlag (2009)
19. Jarecki, S., Lysyanskaya, A.: Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 221–242. Springer-Verlag (2000)
20. Malone-Lee, J., Smart, N.P.: Modifications of ECDSA. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 1–12. Springer-Verlag (2002)
21. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall (2003)
22. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract). In: Davies, D.W. (ed.) EUROCRYPT '91. LNCS, vol. 547, pp. 522–526. Springer-Verlag (1991)
23. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO '91. LNCS, vol. 576, pp. 129–140. Springer-Verlag (1992)
24. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. J. of Cryptology 13(3), 361–396 (2000)
25. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO '89. LNCS, vol. 435, pp. 239–252. Springer-Verlag (1989)
26. Schoenmakers, B.: A simple publicly verifiable secret sharing scheme and its application to electronic voting. In: Wiener, M.J. (ed.) CRYPTO '99. LNCS, vol. 1666, pp. 148–164. Springer-Verlag (1999)
27. Shamir, A.: How to share a secret. Comm. of the ACM 22(11), 612–613 (1979)
28. Smart, N.P., Vercauteren, F.: On computable isomorphisms in efficient asymmetric pairing-based systems. Discrete Applied Mathematics 155(4), 538–547 (2007)
29. Stadler, M.: Publicly verifiable secret sharing. In: Maurer, U.M. (ed.) EUROCRYPT '96. LNCS, vol. 1070, pp. 190–199. Springer-Verlag (1996)